# 函数

函数：

函数是可以重复执行的语句块，可以重复调用，

函数的作用：

用于封装语句，提高代码的重用性

语法规则：

def 函数名(形式参数)：

语句块（代码块）

例如：

def mysum(a,b,c):

d=a+b+c

return d

print(mysum(34,1,2))

函数名和这段代码进行绑定

return 语句：

return 表达式

注意：表达式也可以没有

作用：

1、结束函数的执行。

2、返回调用函数的地方

3、返回一个对象的引用关系到函数调用的地方。

注意：

1、没有return语句或者return语句后没有内容，返回的是None

注意区分一下return 和print 没有任何关系

# 传参

位置传参：

def myfun(a,b,c):

pass

myfun(10, 20, 30)

序列传参：

def myfun(a,b,c):

pass

myfun(\*(10, 20, 30))

关键字传参：

def myfun(a,b,c):

pass

myfun(a = 10,b = 20,c =30)

字典传参：

def myfun(a,b,c):

pass

T={"a":10,"b":20,"c":30}

myfun(\*\*T)

# 缺省参数

函数的缺省参数：

def 函数名（形参名1 = 默认数据1，形参名2 =默认数据2，.......）：

pass

默认传参必须是从右到左，即只要是有默认传参，右边一点是默认传参。

默认传参可以是0个或者是多个，甚至全部都是默认传参。

myfun(a,b,c = 200)

myfun(100,200)对

myfun(100,200,300)对

myfun(a,b=200,c) 错

myfun(100,200,300)对

myfun(100,200)错

函数传参综合使用：

1.函数的传参方式，在能够确认形参唯一匹配的情况下，可以任意组合

# 函数嵌套

1. 函数的嵌套

def语句可以位于任何位置

2. 函数可以赋值给变量,例如：

def my()

print(‘666’)

return 6

调用该函数时，可以不通过函数名称调用，将改函数赋值给变量，通过变量名称进行调用：

f = my

print(f(a,b,c)) 等同于 print(my(a,b,c))

f= my()

3. 函数可以作为参数传递，例如：

def test(a,b,c,func)

return (func(a,b,c)) # 调用func函数，将a,b,c作为参数

print(test(3,4,5,my\_add)) # 在test函数中调用my\_add函数

4. 函数可以作为返回值，例如：

def get\_func():

return my\_add; # 返回函数，而不是变量

使用：

f2 = get\_func(); # f2是指向函数的变量

print(f2(3,4,5)); # 等价于调用 my\_add(3,4,5)

# 变量作用域LEGB

1).本地作用域（L）

局部变量仅仅是暂时的存在，依赖创建该局部作用域的函数是否处于活动的状态。所以，一般建议尽量少定义全局变量，因为全局变量在模块文件运行的过程中会一直存在，占用内存空间。

2).上一层本地作用域（E）

E也包含在def关键字中，E和L是相对的，E相对于更上层的函数而言也是L。与L的区别在于，对一个函数而言，L是定义在此函数内部的局部作用域，而E是定义在此函数的上一层父级函数的局部作用域。主要是为了实现Python的闭包，而增加的实现。

def outer():

a=10

def inner():

b=10

3).全局作用域（G）

在模块文件顶层声明的变量具有全局作用域

全局作用域的作用范围仅限于单个模块文件内

4)内置作用域（B）

max(val1, val2) # max is a built-in fun,

函数也可以赋值给变量，所以所有的内置函数的作用域都可视作B

# lambda表达式

1）用来创建匿名函数，不用return语句

表达式的值就是匿名函数的返回值

2）对匿名函数支持有限，在简单情况下使用

foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]

print map(lambda x: x \* 2 + 10, foo)

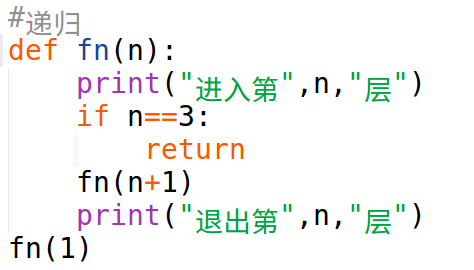
[14, 46, 28, 54, 44, 58, 26, 34, 64]

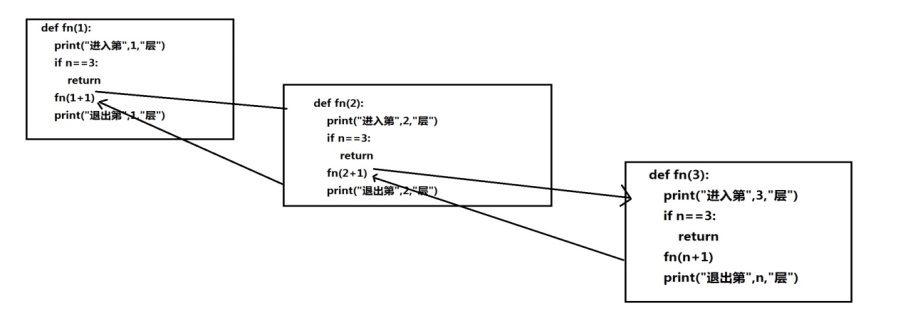
g = lambda x:x+1

g(1)

g(2)

# 递归函数





|  |
| --- |
| 递归函数有层级限制，超过最大层级报错  可以通过sys.getrecursionlimit()查看  可以通过sys.setrecursionlimit()修改  import sys  def fn(n):  if n==1:  return 1  else:  return n\*fn(n-1)  print(fn(998))  sys.setrecursionlimit(1500)  print(sys.getrecursionlimit()) |